Search Tree Pruning for Progressive Neural Architecture Search (Student Abstract)

Deanna Flynn
deannaflynn@gci.net
University of Alaska Anchorage
3211 Providence Dr.
Anchorage, Alaska 99508

P. Michael Furlong
padraig.m.furlong@nasa.gov
NASA Ames Research Center
Moffett Federal Airfield
Mountain View, CA 94035

Brian Coltin
brian.coltin@nasa.gov
NASA Ames Research Center
Moffett Federal Airfield
Mountain View, CA 94035

Abstract

Our neural architecture search algorithm progressively searches a tree of neural network architectures. Child nodes are created by inserting new layers determined by a transition graph into a parent network up to a maximum depth and pruned when performance is worse than its parent. This increases efficiency but makes the algorithm greedy. Simpler networks are successfully found before more complex ones that can achieve benchmark performance similar to other top-performing networks.

Introduction

In this paper, we propose a neural architecture search algorithm that generates a search tree of possible neural networks. This is done by defining a transition graph of valid network layers. The transition graph eliminates the exhaustive search of additional network layers within a network and is the only form of human intervention that our algorithm requires. For nearly any problem, our algorithm creates a more efficient way of finding simple neural networks (fewer parameters and layers).

Our algorithm is a progressive neural architecture search, derived from Levin search (Schmidhuber 1997). Levin search finds low-complexity networks by starting with simple networks and adds complexity until a suitable network is discovered. The Automatic Statistician conducts a progressive search of kernels for Gaussian Process regression (Duvenaud et al. 2013). Parent-child relationships are defined by the composition of atomic kernel functions. Similarly, we construct neural networks by inserting layers.

This algorithm differs from recent Neural Architecture Search (NAS) algorithms that find cell structures for fixed-depth networks (Liu, Simonyan, and Yang 2018; Zoph and Le 2016; Liu et al. 2017). However, (Liu et al. 2017) uses progressive search to find the cell structure and prunes generated networks that are not unique in architecture. For full optimization of cell-based NAS, the number of network layers must also be optimized. A two-stage optimization can accelerate NAS, but the cell structures favor convolutional neural networks (Zoph and Le 2016). Our approach can find non-convolutional networks.

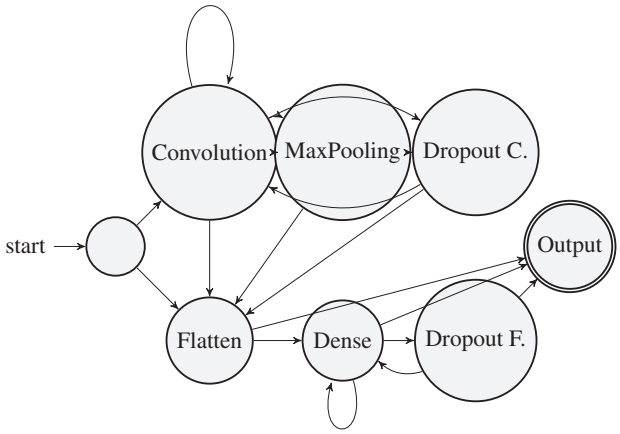

Figure 1: This transition graph defines valid neural network layer sequences in the search tree. *Output* is a 10 neuron *Dense* softmax layer

Algorithm

The algorithm is a Depth First Search (DFS) of a tree of network architectures. The input and output layers are defined by the problem specification. We expand the search tree by adding new layers into leaf-node networks up to a maximum depth. Child nodes inherit the layers, weights, and hyperparameters of their parents. New layers are drawn from transition graph edges leaving the network insertion point.

New layers' hyperparameters are optimized by testing predefined values but this process takes more than half of the runtime. Other optimization procedures, like Bayesian optimization, could decrease execution time.

Child networks are trained for a fixed number of epochs and are pruned if they perform worse than their parent, reducing search time. The pruning rule could exclude optimal networks. However, multiple networks with the same layer sequence, but different hyperparameters, can be discovered within this tree, providing some redundancy.

Central to this algorithm is neural network layer transition graph (Fig 1). We used (*Convolution, Dense, Dropout, Flatten,* and *Max Pooling*) layer types. Transition graph edges were chosen by surveying neural network literature. Net-

Model	# of Parameters	Time to Train	Computer Specification
GoogleNet	4M		CPU
VGG16	∼26M	Weeks	Nvidia Titan Black GPUs
AlexNet	∼60M	6 days	2 Nvidia GeForce 580 GPUs
Our Model	98K	4 days	Intel i7 8th Generation CPU

Table 1: Size and execution time of some Fashion MNIST networks compared to the generated network.

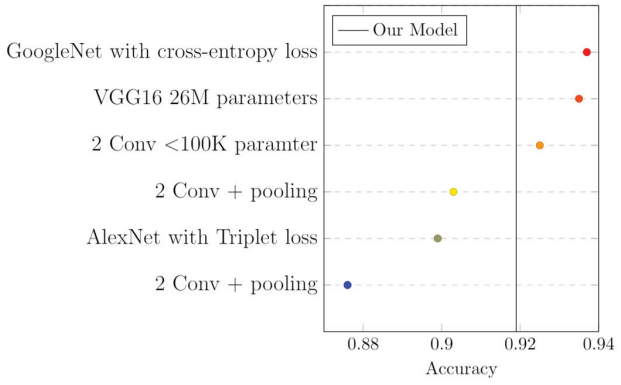

Figure 2: Accuracy of non-preprocessed model vs our model on Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017)

works architectures are paths from *start* to the *Output* node.

Progressively adding network layers provides a Levin-like search of increasing complexity. We continue searching until a maximum depth, but this depth constraint can be replaced with a threshold on network performance. The algorithm should identify simpler networks (fewer parameters) that have desired performance before more complex ones.

Experiment

The algorithm was tested on the unprocessed Fashion-MNIST dataset (Xiao, Rasul, and Vollgraf 2017) with a maximum depth of three. After four days of running on an Intel i7 8th generation CPU, 61 different network architectures were created with the best having an accuracy of 91.9%. Fig 2 compares our algorithm's best accuracy to other networks trained on the same dataset. Our model has $<2\%$ difference in accuracy compared to GoogleNet's accuracy of 93.7% (Xiao, Rasul, and Vollgraf 2017).

Table 1 shows further comparison of GoogleNet, VGG16, and AlexNet against our best model found with Fashion MNIST. GoogleNet, VGG16, and AlexNet each have over four million parameters while the generated model has less than one hundred thousand parameters. No definitive runtime for GoogleNet was found on a CPU, but creators estimate at least one week on a few GPU is needed to train the full network (Szegedy et al. 2015). AlexNet and VGG16 both ran on multiple GPUs, needing 6 days and several weeks to train, respectively (Krizhevsky, Sutskever, and Hinton 2012; Simonyan and Zisserman 2014). Recall our algorithm ran for only four days on a single CPU, generating, training, and evaluating 61 models. Benchmark training times in Table 1 refer to single networks and exclude human expert refining.

Conclusion

Our algorithm progressively searches a tree of neural networks. The algorithm is simple to implement, but depends on the transition graph to define allowable networks. The search space is further restricted by pruning less-performant children, risking missing optimal networks.

Our approach contrasts with cell-based NAS algorithms by optimizing the whole network, and can produce non-convolutional networks. The search space can be tailored through the design of the transition graph, determining what kinds of networks the search algorithm can produce.

We tested the algorithm on the Fashion-MNIST dataset without preprocessing. Training and evaluating networks on a CPU produced a simple model with preformance comparable to benchmarks, but without human intervention. The result was acquired in less time than to train and test other benchmark models on the same dataset.

References

Duvenaud, D.; Lloyd, J. R.; Grosse, R.; Tenenbaum, J. B.; and Ghahramani, Z. 2013. Structure discovery in non-parametric regression through compositional kernel search. *arXiv preprint arXiv:1302.4922*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Liu, C.; Zoph, B.; Shlens, J.; Hua, W.; Li, L.; Fei-Fei, L.; Yuille, A. L.; Huang, J.; and Murphy, K. 2017. Progressive neural architecture search. *CoRR* abs/1712.00559.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. DARTS: differentiable architecture search. *CoRR* abs/1806.09055.

Schmidhuber, J. 1997. Discovering neural nets with low kolmogorov complexity and high generalization capability. *Neural Networks* 10(5):857–873.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. *CVPR*.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Zoph, B., and Le, Q. V. 2016. Neural architecture search with reinforcement learning. *CoRR* abs/1611.01578.